GENERIC OPTICAL ROUTING INFORMATION BASE SUPPORT

[0001] This invention claims the benefit of US Provisional application No. 60/273,641 filed March 7, 2001.

Field Of The Invention:

[0002] This invention relates to communications systems and more particularly to optical routing and signaling for such systems.

Background

[0003] Future agile optical networks are, at the request of customers, expected to quickly and automatically provision lightpaths. Successful provisioning depends on two basic functions of network control: routing and signaling. The routing function automatically updates the optical network topology and related resource information so that a node can compute a light path for the request. The signaling function ensures that the nodes along a computed route exchange the information to set up or tear down a lightpath without user intervention.

[0004] Most current control network approaches are based on the extension of existing Internet Protocols (IP). The IP routing protocol OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System) are extended to exchange optical network routing information and construct the optical routing information database. These extended protocols rely on the instant and periodic exchange of the link state information (LSA {Link State Advertisement} of OSPF, or LSPDU {Link State Protocol Data Unit} of IS-IS) between the directly connected (physically or logically) pairs of nodes, called neighbors. The extension of these routing protocols introduces new optical network related information that should be exchanged using the existing LSA or LSPDU mechanisms. This new information is coded by a number of Type/Length/Value triplets (TLV). For OSPF, these TLVs

are exchanged by Opaque LSAs that are designed to exchange application related information. In the IS-IS protocol, the new TLVs are appended to the LSPDUs. In both cases, the information conveyed by the TLVs is not understandable by the protocol. There is no further processing by the protocol, other than checking the integrity of the data exchange, storing it in an internal format, and notifying interested applications that the information was received.

[0005] In the context of optical networks, the optical link information, such as end point IDs, link type, bandwidth, encoding type, shared risk group IDs, etc., are exchanged by the routing protocols. This information will be used or updated by different applications. For example, the route computation component needs the topology information and the related optical link properties to find a route to set up a lightpath the meets the constraints of the data. The link management applications, which interact with the routing protocol to add or remove an optical link on the node, can also update the information on an existing optical link, e.g., available resources changes.

[0006] There are basically three types of interactions between the applications and the routing engines for Optical Link State (OLS) information processing. The first changes the OLS information, e.g. adds or removes an optical link. In the second interaction the routing engine notifies interested applications that a new OLS is received, an existing OLS is deleted, an existing OLS is updated, etc. In the third interaction other applications use the OLS information for route computation, topology discovery, etc. Changes to the OLS information will cause the OLS information to flood the routing space.

[0007] As described above, all of the OLS information is stored in the internal format in a routing engine without further processing. It is possible that applications can access the OLS in the routing engine, but this type of access is

unacceptable for the following reasons. First, as the OLS is encoded in TLVs, each time an application searches the information it needs to translate the TLVs into some readable data structure; this wastes time and processing power. Second, access to a specific routing engine requires creating applications with protocol-specific code; this limits the flexibility of the network. Additionally, the topology changes in an optical network are much less than in an IP network, so it would be beneficial to reduce the repetition in processing the TLVs.

Summary of the Invention

[0008] Based on the above analyses, this application proposes a solution that makes access to the OLS information much more efficient and free from underlying routing protocols. This proposal applies to all Link State based IGPs (Interior Gateway Protocols) with the extension to optical networks, including OSPF and IS-IS.

[0009] The main goal of this invention is to design a protocol-independent solution that facilitates the update and use of the OLS information. To achieve this goal, we propose a generic interface for applications that manage or use the OLS, and a unified ORIB (Optical Routing Information Base).

[0010] With the generic interface, the applications that need to interact with the OLS can be developed without knowledge of the details of the underlying routing protocols. The applications are unchanged if a new routing protocol is deployed into the node, or an existing routing protocol is modified because of changes in the standard.

[0011] The unified ORIB ensures that the OLS information base structure is independent of the data structure specific to underlying routing protocols. This separation gives a system designer the flexibility to choose the appropriate

database design according to his/her system design requirements, without having to take into account the impact of the underlying protocol.

[0012] Therefore in accordance with a first aspect of the present invention there is provided a system for providing access to optical link state information in an optical network comprising: a unified database that maintains optical link state information independent of data structure of underlying routing protocols; and a protocol independent generic interface between the unified database and applications requiring access to the optical link state information.

[0013] In accordance with a second aspect of the invention there is provided a method of providing access to optical link state (OLS) information in an optical network comprising: maintaining optical link state information in a unified database, the optical link state information being independent of data structure of underlying routing protocols; and providing a protocol independent generic interface between the unified database and applications requiring access to the optical link state information.

Brief Description of the Drawings

[0014] The invention will now be described in greater detail with reference to the attached drawing of Figure 1, which illustrates one example of a generic optical routing information base management software architecture.

Detailed Description of the Invention

[0015] An implementation of the solution according to the present invention follows.

[0016] As shown in Figure 1 the solution can be implemented by four key components: a unified ORIB 12, an OLM (Optical Link Manager) 14, a REM

(Routing Engine Manager) 16, and a generic REM interface 18. Figure 1 illustrates the software architecture of the implementation.

[0017] The ORIB 12 is a unified database that hides the details of the underlying routing protocols. The ORIB is implemented as an array of OLSD (Optical Link State Data) structures. The data structure is designed for ease of use by the interested applications. An OLSD contains the properties of the optical link that can be understood by any routing protocol, e.g. OSPF 20 or IS-IS 22. This information is extracted from an OLS and stored in the routing engine. The OLSD has an internal key that maps to the corresponding OLS in the routing engine. This key ensures a one-to-one relationship between an OLSD in the ORIB and a corresponding OLS stored in the routing engine. When an OLS is updated in the routing engine, the corresponding OLSD in the ORIB is also updated. This update keeps the ORIB and routing engine OLS database synchronized.

[0018] The OLM 14 component manages the optical links attached to the local optical interface. The OLM configures, maintains and monitors the optical link departures and terminations on the node.

[0019] The REM 16 manages the underlying routing protocols. It also configures, maintains and monitors the underlying routing engines. The REM can plug multiple routing engines into the system. The REM interacts with the ORIB Synchronizer 24 to update the ORIB when an update is received from the routing engine. The optical link information is exchanged between the REM and the ORIB Synchronizer by means of the generic data structure of the OLS_payload. The OLS_payload is independent of the underlying routing engines. The REM is responsible for constructing the protocol-dependent PDUs (Protocol Data Unit) from the OLS_payload. The REM also interacts with the OLM, via the ORIB

Synchronizer, to propagate the local OLS changes into the network 32 via the underlying routing engines.

[0020] The ORIB Interface 18 offers a variety of ORIB retrieval services to the applications 26, 28, 29. For example, before computing a new route, the Routing Computation application 26 needs to find all the optical links that satisfy the data stream constraints. It then computes a route based on the results of the ORIB search. The Topology Discovery Application 28 asks the ORIB to list all of the optical links so it can draw a network map. This map is updated according to the ORIB information, through the notification services offered by the ORIB interface. The ORIB interface hides the routing protocol details from the higher level applications, so that they can be developed without knowing the ORIB structure and the underlying routing protocol details.

[0021] The ORIB Synchronizer interacts with the REM and the OLM to keep the ORIB synchronized with the OLS database in the routing engine.

[0022] The TLV translator 30 offers two-way translation service. It can translate a given TLV into a readable data structure, and it can translate the information of a readable data structure into a TLV. This translator is designed to be independent of any specific TLV definition. The translator will consult a lookup table that contains all the TLVs supported by the system to do the translation. Changes in the definition of a TLV will not affect the implementation of the TLV Translator; only the lookup table needs to be updated.

[0023] To show how the whole system works to update the ORIB, two examples are given below to illustrate the updating procedure. The first example describes how the system works when OSPF receives an opaque LSA from the network 32.

The second example illustrates how a new optical link created by the OLM will be propagated into the network.

Example 1. OSPF receives an opaque LSA.

[0024] When the underlying OSPF routing engine receives an opaque LSA from the network, it checks the integrity of the LSA, and stores it in its own LSDB (Link State Database). It then sends a notification to the REM telling it that an opaque LSA has been received. The notification contains the generic OLS_payload data structure constructed from the LSA, and an index that points to the LSA in the OSPF LSDB. The REM forwards the notification to the ORIB Synchronizer, which checks the ORIB for the index of the LSA and determines whether this LSA already exists.

[0025] If the LSA already exists, then it needs to be updated. First, the ORIB Synchronizer locks the LSA in the ORIB so that the other applications cannot access it. Then the ORIB Synchronizer extracts the top level TLV from OLS_payload, and requests the TLV translator to translate it into an OLSD structure with the top-level values filled in. If the translation is successful, the ORIB Synchronizer extracts the second TLV, and asks the TLV translator to translate it as well. After the translation, the corresponding data value is filled in the OLSD. The procedure repeats until the last TLV is extracted and translated. After all the TLVs from the OLS_payload are translated, the ORIB checks the integrity of the created OLSD. If the OLSD is correct, the ORIB overwrites the OLSD in the ORIB, and unlocks the LSA. If any failure occurs in the translation or during integrity checking, the LSA will be dropped, and the previous OLSD will be unlocked for access without change.

[0026] If no OLSD in the ORIB is found from the index of LSA, then a new LSA has been received. The translation procedure is the same except that a new OLSD will be inserted into the ORIB after the translation.

Example 2. Propagating a newly created optical link

[0027] Suppose that a new optical link is created by the OLM. The OLM informs the ORIB Synchronizer of the creation of the new optical link by passing an OLSD structure with the index fields set to NULL. The ORIB then asks the TLV translator to translate the OLSD into TLVs. If the translation is successful, the ORIB Synchronizer creates a new OLS_payload data structure, and requests the REM to flood it into the network.

[0028] After receipt of the request, the REM will create an opaque LSA from the OLS_payload data, and ask the underlying routing engine to install it into its LSDB. The REM first launches the OSPF flooding procedure to propagate the new LSA into the network, then, it sends a notification to the ORIB Synchronizer to inform it that an LSA has been received. The ORIB Synchronizer then starts the ORIB synchronization procedure described in Example 1.

[0029] While specific embodiments of the invention have been described and illustrated it will be apparent to one skilled in the art that numerous changes can be made without departing from the basic concepts. It is to be understood that such changes will fall within the full scope of the invention as defined by the appended claims.